# CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTENTS</td>
<td>1</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>3</td>
</tr>
<tr>
<td>List of tables</td>
<td>4</td>
</tr>
<tr>
<td>Appendix A</td>
<td>7</td>
</tr>
<tr>
<td>ORIGINAL CODES VERIFICATION</td>
<td>7</td>
</tr>
<tr>
<td>A.1 Verification of HABIT codes</td>
<td>7</td>
</tr>
<tr>
<td>A.2 Verification of ARCON96</td>
<td>10</td>
</tr>
<tr>
<td>Appendix B</td>
<td>13</td>
</tr>
<tr>
<td>Meteorological concepts</td>
<td>13</td>
</tr>
<tr>
<td>B.1 Introduction to meteorology</td>
<td>13</td>
</tr>
<tr>
<td>B.2 Atmosphere thermal balance</td>
<td>13</td>
</tr>
<tr>
<td>B.2.1 Atmospheric pressure and wind</td>
<td>14</td>
</tr>
<tr>
<td>B.3 Vertical circulation and Atmospheric stability</td>
<td>15</td>
</tr>
<tr>
<td>B.3.1 Vertical temperature gradient</td>
<td>15</td>
</tr>
<tr>
<td>B.4 Instable conditions</td>
<td>15</td>
</tr>
<tr>
<td>B.4.1 Neutral conditions</td>
<td>17</td>
</tr>
<tr>
<td>B.4.2 Stable conditions</td>
<td>17</td>
</tr>
<tr>
<td>B.4.3 Inversion</td>
<td>18</td>
</tr>
<tr>
<td>B.5 Atmospheric dispersion</td>
<td>18</td>
</tr>
<tr>
<td>B.6 Bibliography</td>
<td>20</td>
</tr>
<tr>
<td>Appendix C</td>
<td>23</td>
</tr>
<tr>
<td>Programming methodology for Instantaneous release</td>
<td>23</td>
</tr>
<tr>
<td>C.1 Computational methodology to run MC simulation</td>
<td>23</td>
</tr>
<tr>
<td>C.2 Data pre-processing</td>
<td>23</td>
</tr>
<tr>
<td>C.3 Modules description</td>
<td>26</td>
</tr>
<tr>
<td>C.3.1 Python Loop Module</td>
<td>26</td>
</tr>
<tr>
<td>Appendixes</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>C.3.2 Main Python Module</td>
<td>28</td>
</tr>
<tr>
<td>C.3.3 Python Class-MC module</td>
<td>29</td>
</tr>
<tr>
<td>C.3.4 Batch file for running EXTRAN and CHEM</td>
<td>36</td>
</tr>
<tr>
<td>C.4 References</td>
<td>39</td>
</tr>
<tr>
<td>Appendix D</td>
<td>42</td>
</tr>
<tr>
<td>Programming methodology for Continuous release</td>
<td>42</td>
</tr>
<tr>
<td>D.1 Data pre-processing</td>
<td>42</td>
</tr>
<tr>
<td>D.2 MC computational methodology</td>
<td>44</td>
</tr>
<tr>
<td>D.2.1 Relative concentration calculations</td>
<td>47</td>
</tr>
<tr>
<td>D.2.2 MC calculations</td>
<td>50</td>
</tr>
<tr>
<td>D.3 References</td>
<td>53</td>
</tr>
<tr>
<td>Appendix E</td>
<td>55</td>
</tr>
<tr>
<td>Programming features and issues</td>
<td>55</td>
</tr>
<tr>
<td>E.1 Programming language issues</td>
<td>55</td>
</tr>
<tr>
<td>E.1.1 F2PY and wrapping Fortran</td>
<td>55</td>
</tr>
<tr>
<td>E.1.2 A brief attempt with Julia programming language</td>
<td>56</td>
</tr>
<tr>
<td>E.2 References</td>
<td>57</td>
</tr>
<tr>
<td>APPENDIX F DEFINITION OF CONCEPTS OF NUCLEAR TECHNOLOGY</td>
<td>59</td>
</tr>
<tr>
<td>F.1 NUCLEAR ENERGY AND RADIOACTIVITY</td>
<td>59</td>
</tr>
<tr>
<td>F.2 Description of a PWR nuclear plant</td>
<td>62</td>
</tr>
<tr>
<td>F.2.1 NUCLEAR PLANT DESCRIPTION</td>
<td>62</td>
</tr>
<tr>
<td>F.2.2 OPERATION MODES</td>
<td>63</td>
</tr>
<tr>
<td>ENVIRONMENTAL IMPACT</td>
<td>65</td>
</tr>
<tr>
<td>COST ANALYSIS</td>
<td>67</td>
</tr>
<tr>
<td>1. Resources distribution</td>
<td>67</td>
</tr>
<tr>
<td>2. Labour costs quantification</td>
<td>68</td>
</tr>
<tr>
<td>3. Cost of material resources</td>
<td>69</td>
</tr>
<tr>
<td>4. Total costs table</td>
<td>69</td>
</tr>
<tr>
<td>PROJECT GANTT DIAGRAM</td>
<td>70</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure A.2. Instructions to run the sample case 1 of HABIT (EXTRAN & CHEM) ..........8
Figure A.3. EXTRAN Demo1 output executed by user ..............................................8
Figure A.4. CHEM Demo1 output executed by user .................................................9
Figure A.5. EXTRAN Demo1 output provided by NEA .............................................9
Figure A.6. CHEM Demo1 output provided by NEA .............................................10
Figure A.7. Hardware and software by which the code has been tested ...............10
Figure A.8. User execution of sample 1 .................................................................11
Figure A.8. NEA execution of sample 1 .................................................................11
Figure C.1. Process diagram of MC methodology for instantaneous release ..........25
Figure D.2. Part 1/2 from Loop Python Module .................................................26
Figure D.3. Part 2/2 from Loop Python Module .................................................27
Figure D.4 Batch file ExePy.bat .................................................................27
Figure C.5. Main Python module script ...............................................................28
Figure C.6 Part 1/8 of Monte Carlo Python module. Wind speed generation method ..29
Figure C.6 Part 2/8 of Monte Carlo Python module. Direction generation method .....30
Figure C.6 Part 3/8 of Monte Carlo Python module. Temperature gradient generation method ..................................................................................................................31
Figure C.6 Part 4/8 of Monte Carlo Python module. Point of accident generation method ..................................................................................................................32
Figure C.6 Part 5/8 of Monte Carlo Python module. Method for modifying input file with generated data ..................................................................................................33
Figure C.6 Part 6/8 of Monte Carlo Python module. Method for filtering generated data (part 1/3) ..................................................................................................34
Figure C.6 Part 7/8 of Monte Carlo Python module. Method for filtering generated data (part 2/3) ..................................................................................................35
Figure C.6 Part 8/8 of Monte Carlo Python module. Method for filtering generated data (part 3/3) ..................................................................................................36
Figure C.7. Batch file for running EXTRAN, CHEM and the recording Python file “exeRec.py” ........................................................................................................37
Figure C.8. Reading results method of record class ..............................................38
Figure D.1. Script to get one hour average data, from wind speed, wind direction and temperature gradient .................................................................43
Figure D.2 wrong Fortran input .................................................................44
Figure D.3. Proper Fortran input..................................................................44
Figure D.4. Process diagram of MC methodology in continuous release evaluation.....46
Figure D.4. Part (1/3) Python script to automate ARCON96 calculations on each point of accident. ..........................................................47
Figure D.4. Part (2/3) Python script to automate ARCON96 calculations on each point of accident. ........................................................................48
Figure D.4. Part (3/3) Python script to automate ARCON96 calculations on each point of accident. ........................................................................48
Figure D.5. Batch file to run ARCON96 for each point of accident..............................49
Figure D.6. Script to record the maximum concentration for each time-period and each point. ........................................................................................50
Figure D.7. Description of ARCON96 output ......................................................52
Figure D.8. MC method for obtaining relative concentration from ARCON96 output ..52

LIST OF TABLES

Table 1 Pasquill-Gifford dispersion parameters ..................................................35
APPENDIX A

ORIGINAL CODES VERIFICATION
Appendix A
ORIGINAL CODES VERIFICATION

Any validated code that is to be applied in nuclear safety assessment must be verified before being used. The verification consist of assuring that the hardware requirements are met, and that the user understands the code.

A.1 VERIFICATION OF HABIT CODES

The hardware requirements to apply HABIT code calculations consist of Windows XP of 32 bits processor. The computer used in this study has the following features.

<table>
<thead>
<tr>
<th>Windows edition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 8.1 Pro</td>
</tr>
<tr>
<td>© 2013 Microsoft Corporation. All rights reserved.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor:</td>
</tr>
<tr>
<td>Installed memory (RAM):</td>
</tr>
<tr>
<td>System type:</td>
</tr>
<tr>
<td>Pen and Touch:</td>
</tr>
<tr>
<td>Not Available</td>
</tr>
<tr>
<td>Not Available</td>
</tr>
<tr>
<td>32-bit Operating System, x64-based processor</td>
</tr>
<tr>
<td>No Pen or Touch Input is available for this Display</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Computer name, domain and workgroup settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer name: NERVG</td>
</tr>
<tr>
<td>Full computer name: NERVG</td>
</tr>
<tr>
<td>Computer description:</td>
</tr>
<tr>
<td>Workgroup: WORKGROUP</td>
</tr>
</tbody>
</table>

The minimum update requirement seems to be complied. Furthermore, the processor has 32 bits. Then the package is installed and tested.

The first code verified is EXTRAN with a demo input file. Then its output is executed by CHEM. The HABIT package provides with an instruction file to follow the procedure.
Instructions to run sample case 1:
- Start HABIT by double clicking on HABIT.BAT after HABIT and the DEMO files were installed to drive C:\
- Click on "Edit Design" or press "Alt+w"
- Click on "Use Existing Design" (Alt+u)
- Change to folder C:\HAB\DEMO\DEMO1 and select "DEMO.DSG"
- Click on "OK" and afterwards on "Done"
- Click on button EXTRAN (Alt+x)
- Click on button Run (Alt+r)
- When the computations finished press H to return to HABIT
- Click on button CHEM (Alt+m)
- Click on button Run (Alt+r)
- When the computations finished press H to return to HABIT

The results of these two runs are sent to the folder C:\HAB\DEMO\DEMO1

Figure A.2. Instructions to run the sample case 1 of HABIT (EXTRAN & CHEM)

The sample case is about a train accident that releases chlorine. With the application of EXTRAN and CHEM codes it is estimated the concentration in the control room.

The result are shown in the following plots:

Figure A.3. EXTRAN Demo1 output executed by user
Now that the evolution of the first minutes of the exposure in the control room is obtained. It is compared with the output files provided by NEA.
Even though in the EXTRAN output plot seems to be a little difference between NEA and user values of concentration, the final results of CHEM verify the proper operation of HABIT codes in the computer used.

A.2 VERIFICATION OF ARCON96

ARCON96 is verified with the same procedure than HABIT codes are. The code has been tested by the following hardware:

```
PROGRAM-NAME: ARCON 96
PACKAGE-ID: 0811189/001

STATUS: Tested

TESTED ON
- COMPUTER : NEC Powermate CT Intel Pentium III 800MHz
  - TARGA VISIONARY WS1300 AMD Athlon 1200MHz
  - DEC ALPHA 500MHz
- OPERATING SYSTEM : MS WINDOWS 2000 Professional (US Version)
  - MS Windows XP Home Edition (German Version)
  - DEC UNIX OSF1 V4.0 1225
- COMPILER : Lahey Fujitsu FORTRAN IF95 V5.5
  - Compaq Visual Fortran 6
  - DEC F77 V5.2-171-42BB
```

7 Figure A.7. Hardware and software by which the code has been tested
The computer and OS for this purposes are the same described in figure A.1. The sample 1 is executed by user and the result is shown in the following figure.

8 Figure A.8. User execution of sample 1

9 Figure A.8. NEA execution of sample 1

It can be appreciated that the 95th percentile of relative concentration values \((\chi/Q)\) coincide in user and NEA outputs. Therefore, the code is verified.
APPENDIX B

METEOROLOGICAL CONCEPTS
Appendix B
Meteorological concepts

B.1 INTRODUCTION TO METEOROLOGY

The study of the movement of the atmosphere related to climate and weather constitutes Meteorology science. For this type of motion, the discrete molecular nature can be ignored, and the atmosphere can be regarded as a continuous fluid medium. Thus, space is divided in parcels in order to characterize the state of the atmosphere in physical quantities. These physical quantities (density, temperature, pressure, etc...) are assumed to have unique values and their derivatives are assumed to be continuous functions of space and time.

The fundamental laws of fluid mechanics and thermodynamics, which govern the motions of the atmosphere, might be expressed in terms of partial differential equations involving the physical quantities as dependent variables and space and time as independent variables.

Nevertheless, these set of partial differential equations is extremely complex, no general solutions are known to exist. Hence, systematic simplification of governing equations is needed to understand the physical role of atmospheric motions. These simplifications are accomplished by the phenomena observation of weather and climate.

B.2 ATMOSPHERE THERMAL BALANCE

In most of the atmospheric motions, the energy consumed comes from the sun. This energy is transferred through radiation heat as electromagnetic waves. The amount of solar radiation is called insolation or incoming solar radiation. This insolation is determined by four factors:

- Solar constant
- Atmospheric transparency
- The daylight period length
- The incident angle of sun rays.
Solar constant stands for the average amount of received radiation in a point located outside of the atmosphere with an equal distance from both the sun and the earth. The real amount of received radiation varies slightly depending on the energy yielded and the distance between the sun and the earth. However, the other three factors are much more influential in real received insolation compared to solar constant.

The atmospheric transparency depends on how different substances suspended in atmosphere absorb or reflect energy, and it stands for the amount of radiation that penetrates the atmosphere and impacts on the surface of the earth without consumption.

Finally, the daylight period length and angle of sunrays incidence have a big influence on the amount of insolation received, and they depend on the seasonal period.

Thermal balance has to do with the equilibrium between the sun energy entering the atmosphere and the radiation emitted by the earth. It is estimated that, 51% of energy entering the atmosphere is absorbed by the earth, 19% by the atmosphere and 30% is reflected as short wave radiation. The 70% absorbed by earth and atmosphere is reflected finally as long wave radiation.

The variety of material composing earth surface combined to the water surface extension, produces what is known as differential warming. This causes atmospheric circulation, due to pressure disturbances.

B.2.1 ATMOSPHERIC PRESSURE AND WIND

The atmospheric circulation consists of masses of air moving from higher pressure zone or system to a lower pressure system. As this phenomenon takes place, wind is generated, and it is influenced significantly by the presence of different friction forces on the surface. Wind contributes to heat transport among other atmosphere effects on earth, and it is termed as the direction from which it blows. Therefore, a “North wind” would blow from the North to the South. When winds blow within a higher frequency direction, they are called prevailing winds. However, when wind tends to be stronger in one direction, it is called dominant wind. A region prevailing and dominant winds are often affected by global patterns of movement in the atmosphere.

Wind speed increases, as the height does, whereas it decreases, as it get closer to the ground due to friction forces. Furthermore, wind does not consist of a constant flow or current, it is formed by a compound of wind gusts instead. Wind gusts are caused by
surface irregularities, and they generate eddies. These, among other form of turbulence, contribute to heat, humidity and dust movement in the higher air.

B.3 VERTICAL CIRCULATION AND ATMOSPHERIC STABILITY

Vertical circulation can be related to forces between high and low pressure systems, to the raise of an air mass due to terrain or weather front collision, as well as to the convection phenomena.

B.3.1 VERTICAL TEMPERATURE GRADIENT

Vertical temperature gradient stands for the change in temperature of the air, as it gets higher. During normal conditions, the gradient is supposed to be approximately between $6^\circ C$ and $7^\circ C$ per km within the troposphere. However, it might change a lot depending on location and day time. A decrease in temperature with height is defined as a negative vertical gradient, whereas an increase in temperature with height would be a positive one.

How atmosphere behaves as air masses move vertically, depends on atmospheric stability. An stable atmosphere tends to resist against vertical circulation, trapping the confined air in lower layers. The degree of stability may be determined by the temperature difference between a parcel of air and air surrounding it. During unstable conditions, the parcel of air tends to constantly move in vertical direction, from top to the bottom or vice versa. When extreme stable conditions occur, colder air gets trapped by a warmer layer of air located above by. This is called inversion, and it prevents vertical circulation, increasing contaminants concentration in the dispersion models.

B.4 INSTABLE CONDITIONS

During instable conditions a parcel of air that is increasing its height will get colder by the dry adiabatic gradient until it reaches its dew point temperature. At this point, it will get colder by the wet adiabatic vertical gradient. This means that the surrounding atmosphere has a higher vertical gradient than the adiabatic, as can be seen in figure 1. Hence, the parcel of air moving upwards is still warmer than surrounding air, and the temperature difference between gradients increases, therefore increasing the buoyancy.
As air parcel gets higher, colder air masses move below by and repeat the cycle, producing an increase in the vertical circulation. The difference between environmental vertical gradient and dry adiabatic vertical gradient influence on the degree of instability. The following figure 2 shows a comparison between gradients in slightly instable conditions and very instable conditions.

**Figure 1. Buoyancy increase due to instability**

**Figure 2. Instable conditions**
Most common instable conditions occur during sunny days with calm winds and strong insolation. Earth absorbs heat fast and transfer part of it to the surface layer of air. Then, the air mass gets warmer and ascends.

**B.4.1 NEUTRAL CONDITIONS**

When environmental temperature vertical gradient is the same than the dry adiabatic, the atmosphere reaches neutral stability. This neutrality means a threshold that separates stable and instable atmospheric conditions. It normally happens during windy or cloudy days.

**B.4.2 STABLE CONDITIONS**

When environment vertical gradient is lower than the dry adiabatic, the air masses resist against vertical circulation. The ascending air will get colder than its surrounding and, therefore, denser, tending to go back to its original elevation. Stable conditions occur normally during night, with little wind presence.

*Figure 3. Stable conditions*
B.4.3 INVERSION

Inversion occurs when air temperature increases with height. This condition generally takes place in shallow air layer, affected by terrain shape. Furthermore, the parcel of air confined does not tend to disperse when wind blows. Dispersion plumes get trapped in the layer they are emitted, increasing toxic concentrations.

![Inversion temperature graph](image)

**Figure 4. Inversion temperature**

B.5 ATMOSPHERIC DISPERSION

The degree of atmospheric stability has an important effect on the plume dispersion behaviour and the toxic concentrations. Instable conditions will produce looping plume, which facilitates dispersion and decreases concentrations.

On the other hand, a fanning plume takes place when stable conditions occur. Inversion gradient prevents vertical motion and the plume can extend several km downwind. With neutral conditions, however, the plume tends to be conic, as it can be appreciated in figure 5.
Finally, during inversion conditions, the plume can get stacked either below or above the inversion point.
B.6 BIBLIOGRAPHY


TOXIC GAS DISPERSION CONCENTRATION AFFECTING A NUCLEAR POWER PLANT CONTROL ROOM.
APPENDIX C

PROGRAMMING METHODOLOGY FOR
INSTANTANEOUS RELEASE
Appendix C
Programming methodology for Instantaneous release

This section contains a description of programming processes and methods that lead to the Monte Carlo simulation implementation, in the case of instantaneous release.

C.1 COMPUTATIONAL METHODOLOGY TO RUN MC SIMULATION

As it is commented in Appendix E to the main document of this study, the chosen programming languages to iterate dispersion calculations are Python™ and Batch. Python™ is used for the methods that process data and generate random variables. Batch files are used to call Fortran original codes. Moreover, Batch file has been also used to separate looping Python™ module from the whole cycle of a history in order to prevent bottleneck effect. This effect takes place when the MC iteration calls another module, for instance the Main module, and keeps iterating without waiting for the other process to finish. Then, the tasks run overlapped preventing the modification of input files, result files, control files, etc. Since the modules do not have time to modify stored data, the dispersion codes execute every time the same inputs returning the same concentration repeatedly.

For this reason, modules separation and result control have become key points in the design of this application. In the figure D.1 can be appreciated the process diagram for the MC methodology for instantaneous release.

C.2 DATA PRE-PROCESSING

The random variables generation is performed to emulate meteorological conditions, as well as the point of release, which is also referenced in this study as the point of derailment\(^1\). The point of derailment is distributed in a uniform way for execution

\(^1\) The most accurate name would be the point of release. However, after a brief study of train accident phenomenology, it could be said that, in most of the cases, the cause of a toxic release will be the train derailment. This hypothesis is supported by the fact that case study area has little chances of another kind of accident but derailment.
purposes. Once the statistical study about trains accident is finished (in further studies), the probabilistic results are introduced with a cumulative distribution function shape, which will be applied in the MC process.

After the statistic study about meteorological conditions is done, the results are presented as cumulative distribution functions of wind speed, direction a temperature
10 Figure C.1. Process diagram of MC methodology for instantaneous release
gradient, which consist of dependent variables. Hence, the first value simulated would be wind speed, so that the result is used to calculate the other two variables. This will be explained at incoming chapter in more detail.

C.3 MODULES DESCRIPTION

The first module described is Python™ Loop Module, which is the module that repeats the whole process of one history, in the MC method, a number of times that returns a solid statistical amount of results.

C.3.1 PYTHON LOOP MODULE

This module starts the MC simulation and produces the iteration of the histories in the MC simulation. The first task is to call the files Python module, to create some data files that will be used during the process. The it starts the loop and, as the first task in the loop, calls Batch file ExePy.bat that will trigger the Main Python Module, initializing a new history. The script can be explored in the following figure.

```python
5  #author: Antonio Gomez
6  ###
7  # ###########################################################
8  #  NERG - UPC , May 2016
9  #  Python Module to iterate MC histories
10  # ###########################################################
11  import files
12  import subprocess,os,sys,time
13  from itertools import count
14  tic=time.clock()
15  print tic
16  def goto(linenum):
17      global line
18      line = linenum
19      path=os.getcwd()
20      u=0
21      for i in count(0):
22          u+=1
23          subp=subprocess.Popen(['ExePy.bat'],shell=True)
24          subp.communicate()
25          time.sleep(0.5)
```

Figure D.2. Part 1/2 from Loop Python Module
The *time.sleep* method is applied to give some margin to the main process before continuing the iteration process.

```
while line==999:
    try:
        checking=open(path+"/loopinGood.txt",r"
        for line in checking:
            valor=int(linea)
            loops+=1
            print ("--------------")
            print ('loop '+str(loops))
            print ("--------------")
            print ('cuentas = '+str(valor))
            print ("""
            print ('BigLoop = '+str(u))
            print ("--------------")
        if loops>1000:
            checking.close()
            raise SystemExit
        elif valor<u:
            checking.close()
            time.sleep(0.1)
            goto(999)
        elif valor==u:
            checking.close()
            break
        elif valor>u:
            checking.close()
            raise SystemExit
    except:
        continue
    if i>501:
        print ('sim num '+str(i))
        break
```

**12 Figure D.3. Part 2/2 from Loop Python Module**

The key point in this script is the creation of a checking result file, *LoopinGood.txt*, with which is possible to control the rhythm of the cycle. This checking file contains the number of finished histories, and it is modified with the last task of the cycle. Then, at the line 33 of this module, the number in *LoopinGood.txt* is checked, and it is compared with the number of iterations done by the Main loop (line 44 to 54). Then, while the number of iterations is less than the number of finished histories, this little loop inside the main loop keeps iterating until the line 51 == True. Then a new history can start.

The Batch script to trigger Main program module does not need much explanation:

```
@echo off
start "python" "c:\HABIT\MainMC.py"
exit
```

**13 Figure D.4 Batch file ExePy.bat**
C.3.2 MAIN PYTHON MODULE

Main module starts a new history by assigning a class for the MC class module. The new history is then an object belonging to the MC class. The attributes consist in most cases of files that contain data or are used to save information. The method `time.sleep` is also useful here to allow different methods to handle these files and to close them before the next method call them.

```python
import os, time, subprocess
from ClassMC import HistMC
from recResults import record

path=os.getcwd()
History=HistMC()
History.chooseWind("/WindCDF.csv","/velocity.csv")
time.sleep(0.1)
History.chooseDir("/velocity.csv","/DirectionCDF.csv")
time.sleep(0.1)
History.chooseST("/velocity.csv","/GTCDF.csv")
History.choosePoint("/pointCDF.csv","/datapoint.csv","/GeoData.csv")
History.Filter("/datapoint.csv","/velocity.csv","/gausscoeffs.csv")
time.sleep(0.3)
consult=open(path+"/decision.txt","r")
for a in consulta:
    if int(a)==1:
        History.inpModif("/DEM01EX.INP","/datapoint.csv","/velocity.csv","/inpEx1.inp","/inpEx2.inp")
time.sleep(2)
    else:
        subprocess.Popen(['runExtrn.bat',shell=True])
        subprocess.communicate()
    elif int(a)==0:
        Hist2=record()
        Hist2.recording("/zercen.csv")
        Hist2._control("/loopinGood.txt")
    consult.close()
```

Figure C.5. Main Python module script.

After executing the MC methods, there is one method to decide whether to run dispersion models or to go directly to results. This is Filter method and it is used to calculate when the meteorological conditions are favourable to reach concentration levels at the intake of the control room.

Here in the Main module, the filtering is checked by opening a file where the decision is recorded. Depending on the value in the file, the main program triggers dispersion codes (line 27) or goes to finish the process (line 32).
C.3.3 PYTHON CLASS-MC MODULE

In this module the class “ClassMC” is defined. The methods in this class are designed to perform the Monte Carlo simulation, and to filter its results. This module or sub-process is written in an object oriented way. By doing this, the tasks of creating, removing and modifying files to run, as well as directories, eases the path for the user.

The first method consist of choosing wind speed. This is performed by generating random variable and introducing it in the empirical CDF of the wind speed. This CDF is contained in a specific file called MeteoFile.

```python
#author: Antonio Gomez
#
# ***************************************************
# MC method to choose Wind Speed, Wind Direction and
# Temperature Gradient by the CDF extracted
# from the Plant Meteorological Station records
# ***************************************************

class HistMC():
    import os
    import random

    # Method for obtaining wind speed
    # ---------------------------------
    path=os.getcwd()
    roll2=random.uniform(0.0, 0.99999)

    def chooseWind(self, windCDF, velocidad):
        import csv
        self.aborc=open(self.path+windCDF,"rb")
        self.lctor=csv.reader(self.aborc)
        self.escribe=open(self.path+velocidad,"w")
        for row in self.lctor:
            if self.roll2<=float(row[1]):
                break
        self.escribe.write(row[0])
        print (row[0])
        self.aborc.close()
        self.escribe.close()
```

15 Figure C.6 Part 1/8 of Monte Carlo Python module. Wind speed generation method

The resulting wind speed value is saved in the “velocidad” file. Which name will be specified by Main module when the attributes are designed.
As it can be appreciated, wind direction is generated by random variable, but it applies different cumulative distribution function depending on the wind speed is recorded in the file “velocidad”. These CDFs are stored in a csv file, so the number in brackets by each row[#] (lines 48, 51, 54, 57, 60 and 63) stands for columns in the csv file that contains the cumulative distribution functions.

16 Figure C.6 Part 2/8 of Monte Carlo Python module. Direction generation method
In the figure D.6 part 3/8 the method for obtaining the temperature gradient by random variable generation is shown. As it can be appreciated, the procedure is the same as the wind direction generation.
With this method the geometrical parameters are generated. The points are located with a uniform distribution, and they are stored in the Geological file. This file contains information for each point of distance to intake, direction from source to plant and cross-sectional area for building wakes corrections. Therefore, when the random number is generated, it assigns a row in the Geo-file and takes the Geo-values saving them in another file that will be used to modify EXTRAN input.

```python
# Method for obtaining distance to intake, direction
# and building area

point=0
dist=0
roll5=random.uniform(0.0,0.99999)

def choosePoint(self,pointCDF,pointfile,geofile):
    import csv
    self.abre6=open(self.path+pointCDF,"rb")
    self.lector6=csv.reader(self.abre6)
    for row6 in self.lector6:
        self.point+=1
        if self.roll5<float(row6[0]):
            break
    print (self.point,row6)
    self.abre6.close()
    self.abre7=open(self.path+pointfile,"w")
    self.abre8=open(self.path+geofile,"r")
    for value in self.abre8:
        self.dist+=1
        if self.dist==self.point:
            break
    self.abre7.write(value)
    self.abre7.close()
    self.abre8.close()
```

Figure C.6 Part 4/8 of Monte Carlo Python module. Point of accident generation method
The modification of EXTRAN input has the issue of being a Fortran77 input format. This means that the position of the new values has to be exactly the specified in the Fortran subroutine. To achieve that, both the values and the original lines are converted to lists, then the new value substitutes the original file. Depending on its length, it will modify different length of the original input file, as it is seen in lines 156 to 168.

The procedures for modifying Wind speed, Atmospheric stability, Intake height and Building area are the same as the one in figure D.6 part 5/8 for modify Distance.
The filtering is performed taking into account the wind direction generated and the direction from the source to the intake likewise generated by the point selection. Furthermore, to apply the decision criterion it must be estimated the lateral dispersion of the plume, considering the distance also obtained in the choose-Point method. Hence, any wind direction that falls in the dispersion window is considered to blow straight towards the intake.

The wind direction that falls out of the dispersion window will reject the dispersion calculations.

The equation modelled in this method to estimate lateral dispersion is:

\[ \sigma_y(x) = \frac{k_1 x}{\left[1 + \left(\frac{x}{k_2}\right)^3\right]^{k_3}} \]  

(3.1)

Where:
X: is the distance from the source to the control room intake

\(k_1, k_2, k_3\): are coefficients provided by Pasquill-Gifford, ref. [1], in the following table:

<table>
<thead>
<tr>
<th>Stability class</th>
<th>(k_1)</th>
<th>(k_2)</th>
<th>(k_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.250</td>
<td>927</td>
<td>0.189</td>
</tr>
<tr>
<td>B</td>
<td>0.202</td>
<td>370</td>
<td>0.162</td>
</tr>
<tr>
<td>C</td>
<td>0.134</td>
<td>283</td>
<td>0.134</td>
</tr>
<tr>
<td>D</td>
<td>0.088</td>
<td>707</td>
<td>0.135</td>
</tr>
<tr>
<td>E</td>
<td>0.057</td>
<td>1070</td>
<td>0.137</td>
</tr>
<tr>
<td>F</td>
<td>0.037</td>
<td>1170</td>
<td>0.134</td>
</tr>
<tr>
<td>G</td>
<td>0.023</td>
<td>1159</td>
<td>0.124</td>
</tr>
</tbody>
</table>

*Table 1 Pasquill-Gifford dispersion parameters*

The stability class has been determined by the “chooseGT” method. The parameters are set to 0.0 and the distance and stability class are used to assign the coefficients value:

```python
for row17 in self.lector17:
    distance=float(row17[0])
    intakedir=int(row17[1])
for row18 in self.lector18:
    stability=row18[2]
    # Uwind=float(row18[0])
    windir=int(row18[1])
for row19 in self.lector19:
    if row19[0]==stability:
        k1=float(row19[1])
        k2=float(row19[2])
        k3=float(row19[3])
self.abre17.close()
self.abre18.close()
self.abre19.close()
beta=1.0+(distance/k2)
sigma=k1*distance/(beta**k3)
```

*Figure C.6 Part 7/8 of Monte Carlo Python module. Method for filtering generated data (part 2/3)*

As it can be seen, in the lines 271 and 272 the equation (3.1) is computed.
Having the value of the lateral dispersion $\sigma_y$, filtering criteria is applied considering the window of the dispersion and the range of the wind direction. The wind direction is collected in $45^\circ$ range, which means the North winds contains all winds with directions from $337,5^\circ$ to $22,5^\circ$. Once the minimum angle between wind direction and direction to intake is determined, the lateral distance is obtained by distance, and is compared with $\sigma_y$ (line 291 and 296).

Finally, the result is saved in the decision making file, which is checked by Main module when deciding whether to go for dispersion calculations.

### C.3.4 BATCH FILE FOR RUNNING EXTRAN AND CHEM

Once the inputs are modified and the filtering goes positive, the dispersion codes are activated. The better way to achieve that is to call them from a Batch file, which interacts with the Fortran code as though it was executing a command prompt. The Batch script is shown in the following figure.
Figure C.7. Batch file for running EXTRAN, CHEM and the recording Python file “exeRec.py”

In line 4 it can be appreciated that batch copies modified input file into the original one so that Fortran executes it properly. The next line Batch indicates extran.exe to execute the input.nfo file. This is a trick to “cheat” Fortran. The reason to do that, is that extran.exe opens a command prompt of 16 bits that asks the user for the name of four files, which are input and output files. Since batch file cannot open a write on the command prompt as if it was the user, it introduces the input and output names with an external file.

The killExtran and killChem loops are designed to allow both programs to run and save the results before next task is activated.

Finally, Batch calls the module exeRec.py that will record the concentration result. This module creates an object of the class “record()”. This class is defined in module “recResults.py” and its methods consist of: Reading the results, Recording them in the results files and Controlling (creates control files used by Loop Module).
As it is explained at the main report, the filtering is done by wind direction. However, there are times in which wind direction is in window with the direction to intake, but no concentration reaches the intake. In this case, EXTRAN might behave in a non desirable way for the program, that is, it returns output with no columns of concentration. This can lead CHEM to not run properly and, therefore, returning no concentration columns as well. This fact may create gaps between the control file and the Loop Module logical control, and the result would be either an infinite iteration or exiting of the system.

In the figure D.8, it is illustrated the reading results method. It copies the CHEM output on a csv file, so that the recording module can seek for the maximum value and save it. If there are no concentrations columns in CHEM output, it writes a csv with zero values, as it is shown in line 19.

In the method recording the obtained concentration is checked, and in the case the value is zero it gets saved in zero results file, otherwise it is recorded in results file.

The control method opens the result.txt, zero-result.txt and no-reaches.txt files and sum the number of recorded values. The returned value will be used by Loop module to control the next iteration initiation.
C.4 REFERENCES

APPENDIX D

PROGRAMMING METHODOLOGY FOR CONTINUOUS RELEASE
Appendix D
Programming methodology for Continuous release

This section contains a description of programming processes and methods that lead to the Monte Carlo simulation implementation, in the case of continuous release.

D.1 DATA PRE-PROCESSING

The first task in this evaluation has been pre-processing meteorological data from the Plant. The main issue here is to achieve the same format that ARCON96 expects from its meteorological input file. Since is written in Fortran77, there are some requirements to do that.

These are the requirements in detail:

“Each record must include the day of the year, the hour of the day, the stability, and the wind direction and wind speed at the lower measurement level. The record may also include a five-character location identifier, and a wind direction and speed for the upper level. The format for the records is:”

\[(1x, A5, 3x, I3, I2, 2x, I3, I4, 1x, I2, 2x, I3, I4)\]  [NUREG/CR-6331, ref. [1]]

The number plus the ‘x’ are blank spaces, A5 indicates string of 5 characters and finally, the integers are indicated with a ‘i’ followed by a number which stands for number of digits.

Moreover, the code requires the data to be collected in one-hour periods. Here comes the first issue, the meteorological data file provided by the plant, not only has separated files for each parameter, but it is collected within 15 minutes periods. Therefore, the data must be processed before using it with the code. The reason to give detail about this issue is to show what kind of problems are to be found when dealing with big amount of data and old versions of FORTRAN.

The first task consist of obtaining average values for each hour collected. Since the amount of data surpass 23e3 rows, there is no sense in doing it with a worksheet application. Hence, a script is been written for this purpose. The collected data is first joined in a CSV file, and then processed by the Python script showed in the next figure.
Figure D.1. Script to get one hour average data, from wind speed, wind direction and temperature gradient
There are two interesting things about this script. The first one is that not all the rows contained data, so the procedure “try” and “except” prevents the script to return error. The second one is that the average must be done for each three values, in which is distributed each hour data. So the reminder ‘%3==0’ plus the ‘continue’ allows the script to start all over again saving the averages first.

Now the data will be joined again with a worksheet app, and is copied to a text file. Another issue appears at that stage, that is, the characters must be located at the end of the each position determined by Fortran code. So a file that looks like...

```
1 | EXMET 87 1 9999999 99 9999999
2 | EXMET 87 1 21 106 6 44 113
3 | EXMET 87 1 15 40 6 32 39
4 | EXMET 87 1 11 64 6 28 58
5 | EXMET 87 1 9 19 5 23 19
```

...Will not work. It must have this format instead:

```
1 | EXMET 87 1 9999999 99 9999999
2 | EXMET 87 1 106 21 6 113 44
3 | EXMET 87 1 40 15 6 39 32
4 | EXMET 87 1 64 11 6 58 29
```

Actually they are pretty similar, but the first meteo-input does not work, and the reason is not only the fact that wind speed and direction are in the wrong position, but because they are located on the wrong columns. This issue carries some work too. Before processing it again, the data is copied in CSV format. Finally the input file is built with another Python script.

### D.2 MC COMPUTATIONAL METHODOLOGY

In contrast to the methodology applied for instantaneous release, in the case of continuous release the relative concentrations are calculated for each point of accident before launching Monte Carlo simulation. The results are stored in output files and they are be used by the MC modules to generate the possible combination of point of accident, release period and relative concentration.

The process diagram in following figure shows how the computational tools are structured for this purpose.
TOXIC GAS DISPERSION CONCENTRATION AFFECTING A NUCLEAR POWER PLANT CONTROL ROOM.
Figure D.4. Process diagram of MC methodology in continuous release evaluation
D.2.1 RELATIVE CONCENTRATION CALCULATIONS

Once the Meteorological data is ready, the modification of ARCON96 input file is performed for each point of accident. Since this calculation is to be done one time for each point, there is created one input directory for each point, and the inputs are stored there. The modifications consist of Wind speed, direction, distance and cross-sectional area. The script is shown next.

```python
import os
import subprocess
import csv

# **------------------------------------------------------------------------**
# # [~~~~~~~~~~~~~~~~~~~~~~~~[~~~~~~~~~~~~~~~~~~~~~~~~[~~~~~~~~~~~~~~~~~~~~~~~~]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
# # NERG-UPC  april 2016
# # Python version for automation of ARCON96 calculations
# # [~~~~~~~~~~~~~~~~~~~~~~~~[~~~~~~~~~~~~~~~~~~~~~~~~[~~~~~~~~~~~~~~~~~~~~~~~~]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
# # **------------------------------------------------------------------------

# In case that there are input and output files, *******
# it removes them, with the following lines *******
#
if os.path.isdir('inputs') or os.path.isdir('outputs'): try:
    print ('deleting inputs directories...')
    subp=subprocess.Popen(' rmInpOut.bat',shell=True)
    subp.communicate()
except:
    print ('can`t find them')

# Once the directory is cleaned up it starts running ******
# Create inputs folder to generate input files ******
#
path2=os.getcwd()
os.makedirs(path2+'/inputs')
abre2=open(path2+'/genfile.csv','rb')
leascv=csv.reader(abre2)
```

Figure D.4. Part (1/3) Python script to automate ARCON96 calculations on each point of accident.
```python
for row in lines:
    num=num+1
    name='ArcInp'+str(num)+'.inp'
    result=open(path2+'/inputs/'+name,'w')
    abre=open(path2+'/arconin.rsf','r')

    # initiates loop for substituting distances and ************
    # generates inputs files ******** ******************
    # <a> stands for each line in file arconin.rsf

    if 'Distance' in line:
        dista=list(lines)
        b=list(str(int(row[0])))
        if len(b) == 5:
            dista[1:6]=b[0:5]
        elif len(b) == 4:
        elif len(b) == 3:
            dista[3:6]=b[0:3]
        elif len(b) == 2:
            dista[4:6]=b[0:2]
        linea=str(''.join(dista))

        if ' '.join(b) in linea:
            linea='outputs\output'+str(num)+'\ex'+str(num)+'_NEA_user.cfd\n'

        elif 'Building' in linea:
            build=list(lines)
            g=list(str(row[2]))
            if len(g) == 3:
                h=[' ',' ',' ']
            else:
                h=[' ']
            k=h+g
            build[2:8]=k[0:6]
            linea=str(''.join(build))

        elif 'Direction' in linea:
            direc=list(lines)
            c=list(str(int(row[1])))
            if len(c) == 3:
                f=[' ']
            elif len(c) == 2:
                f=[' ',' ']
            elif len(c) == 1:
                f=[' ',' ',' ']
            d=f+c
            direc[0:4]=d[0:5]
            linea=str(''.join(direc))

        result.write(linea)
    abre.close()
result.close()
abre2.close()
sbp=subprocess.Popen(['runinputs.bat',shell=True])
sbp.communicate()
```

30 Figure D.4. Part (2/3) Python script to automate ARCON96 calculations on each point of accident.

31 Figure D.4. Part (3/3) Python script to automate ARCON96 calculations on each point of accident.
By opening the Geo-file Python gets the data for each point and modify an input file which is exactly like the original for ARCON96. First creates inputs directories and store the created files. Once the input data files are created, Python calls the Batch file “runininputs.bat”. This file is been designed to get each modified input file, to substitute the original and to run ARCON96 with its normal procedure. The Batch script is shown in the following figure.

```
@echo off
rem ************************************************************
rem               NERG-UPC april 2016
rem     run generated inputs and locate them
rem in output directories n files
rem ************************************************************
set i=1
:loop
if %i% == 47 goto :eof
mkdir outputs
mkdir outputs\output%i%
copy inputs\ArcInp%i%.inp arconin.rsf >nul:
copy ex1_96nea_user.qa outputs\output%i\ex1_96nea_user.qa
set /a i=i+1
goto :loop
del ex1_96nea_user.qa
pause
```

32 Figure D.5. Batch file to run ARCON96 for each point of accident

It can be appreciated in lines 12 and 13 that Batch calls ARCON96 by its original input file, previously copied from a modified one.

With the calculations finished, the files are ready to be run in the MC simulation. However, some testing analysis is done to check whether the results confirm a proper execution.

The following script is written to record the maximum values of relative concentration for each time period of each point.
Appendixes

The results indicate that the sector of study is too conservative. This sector is determined by an 8 km radius circumference, but the points that are returning relative concentrations at the intake of the control room are closer than 1.5 km to the intake.

D.2.2 MC CALCULATIONS

The modules included in the second step of the process diagram in the figure D.4 are very similar to the MC modules of instantaneous release type. The main differences are, in first place, that only CHEM is being executed, and in second place, the CHEM input has to be adapted with the estimated concentrations.

These estimated concentrations are calculated by the following equation:

\[ \text{Concentration} = \frac{Z}{Q} \cdot \text{Release Rate} \quad (D.1) \]
Where:

\[
\frac{X}{Q}
\]

Stands for relative concentration provided by ARCON96

Release Rate= is calculated for each period considering the amount of material released and flashed (see main document of this project)

These calculations are included in the methods for generation of concentration conditions belonging to the ARCON MC-class module. Then the “EXTRAN” output is modified with new estimated concentration coming from ARCON96 outputs, and it is introduced in CHEM process by the corresponding Batch file.

One interesting feature of this MC class module is how the relative concentrations are chosen from ARCON96 output files. With the next figure, an output is shown and described to ease the understanding of the concentration selection method.
34 Figure D.7. Description of ARCON96 output

In the figure D.7 it can be appreciated that the MC first choose a release period, which can vary from 1h to 720h, and then, for the cumulative distribution function of the chosen period it chooses a frequency of occurrence. However, these CDFs have different top values since they are not standardized to (0,1). So the script must generate a different random for each period. In the following figure it is shown a piece of the method that makes the MC for concentration.

```python
# calculating concentration for chosen period
abrc6=open(self.path+"/topvalue.csv","rb")
lector6=csv.reader(abrc6)
self.abrc5=open(self.path+arccfile,"rb")
self.lector5=csv.reader(self.abrc5)
abrc7=open(self.path+"/roccvalue.txt","w")
abrc66=open(self.path+"/saverate.txt","r")
num=0
for a in abrc66:
    num+=1
    if num==1:
        rate=int(a)
    elif num==2:
        period=int(a)
    elif num==3:
        periodo=int(a)
abrc6.close()
for row3 in lector6:
    roll2=random.randint(0.0,int(str(row3[period])))
print roll2
# print(str(row3[1]))
cont2=0
for row2 in self.lector5:
    #abrc5.write(row)
    cont2+=1
    if cont2<=1:
        continue
    elif int(row2[period])>roll2:
        conci=float(row2[0])*rate
    abrc7.write(str(conci))
    print (str(conci))
    print (str(conci)+"*"+str(periodo))
    break
```

35 Figure D.8. MC method for obtaining relative concentration from ARCON96 output

As it is seen in the line 86, the random number is generated for a top-value that is different for each period. In line 96 the relative concentration is assigned and it is multiplied by the corresponding release rate of the period.

Apart from that, there is no big difference in the MC modules regarding to instantaneous release type.
D.3 REFERENCES

APPENDIX E

PROGRAMMING FEATURES AND ISSUES
Appendix E
Programming features and issues

In this section it is explained the reason of the programming tools selection and some features and issues related with them, as well as other alternative programming languages that have been tested.

E.1 PROGRAMMING LANGUAGE ISSUES

As it was explained in previous chapters, dispersion models used in this study are written in Fortran77. Since the codes were provided by NEA, and they are validated for their utilization in any Nuclear Power Plant, they cannot be modified. Furthermore, the compilation takes place as the *.exe file is executed, not leaving any chance of compiling separated Fortran sub-processes by the user. Hence, when MC simulations are to be implemented, original dispersion codes must be called, as they would be by the Command Prompt generated within the Fortran.exe code.

E.1.1 F2PY AND WRAPPING FORTRAN

At the beginning of this stage, the idea was to wrap Fortran77 codes with Python™ in order to automate input modification, as well as Fortran input calling, for all the histories run in the MC simulation.

The procedure consist of importing F2PY.py module, ref. [1], in order to generate secondary Python files from the original Fortran file, which would be compiled and run as Fortran subroutine. However, it is also necessary to import NumPy and SciPy modules. And one of the most effective ways of doing this, in Windows environment, is installing the package Anaconda2, ref. [2].

Once the package is installed, it is important to check if all meta-programming related to modules needed are suitable for the Python version installed. Here was an issue, because the version of Python installed for this study is v.2.7, and, unfortunately, some necessary modules to run F2PY have v.3.4 expressions and built-in functions. The consequence is, inevitably, a little waste of time trying to modify meta-programming modules. Some users recommend not to use it in Windows environment, but in Linux OS.
At this point, another issue appeared, which was the fact that the original Fortran codes HABIT and ARCON96 are protected against modifications, and the only way of compiling subroutines separately, would be to copy-paste file by file with notepad. In any case, it would be not worthy to modify an already validated code, considering the intended purpose of claiming validation for new methodology. Actually, it would be much more reasonable to build a new code for dispersion calculations from zero, and then, presenting it to the regulator body.

Hence, this option was rejected.

E.1.2 A BRIEF ATTEMPT WITH JULIA PROGRAMMING LANGUAGE

Due to the commented issues, another option is assessed. It consist of using Julia programming language instead of Python. Julia seems to be a promising language, compiled as it runs like Python, but much faster, according to what the developers claim.

Julia was developed in MIT and first appeared in 2012, releasing a stable version in 2015. It uses Just in Time (JIT) compiler that developer claims to be close to C speed of execution.

A little Julia script was tested for the modification of EXTRAN inputs. The operators and command words are similar to that of Python, and its approach to numerical problems reminds Matlab algorithms.

Regarding to Fortran call issue, “Julia has a “no boilerplate” philosophy: functions can be called directly from Julia without any “glue” code, code generation, or compilation — even from the interactive prompt”. Ref. [3]. Therefore, it does not need to wrap Fortran code modifying it and compiling it as a subroutine. Although this feature seemed to suit the purposes of this study, it would not be that easy, however, to put it into practice. The lack of libraries and online support would cost precious time whenever any difficulty had to be faced.

After spending some time trying to call and to control Fortran inputs and outputs from Julia, it was decided to call Batch files instead, and leave the batch file do the input-Fortran call. However, since calling batch files was not a problem for Python, and considering the differences in amount of available libraries and support with Python compared to Julia, it was finally decided to start all over again and to use Python to develop the model. Of course other languages were put on the table, but none of them is that easy and fast to learn and write.
E.2 REFERENCES


APPENDIX F

NUCLEAR TECHNOLOGY CONCEPTS
APPENDIX F
DEFINITION OF CONCEPTS OF NUCLEAR TECHNOLOGY

F.1 NUCLEAR ENERGY AND RADIOACTIVITY

Nuclear energy is the energy that is released spontaneously or artificially in nuclear reactions. In these reactions, the nuclei undergo transformations induced by the interaction of nuclear particles of all kinds (neutrons, protons, alpha, heavy nuclei, etc.) resulting in new nuclides, generally unstable, and therefore radioactive, and the emission of nuclear particles.

Radioactivity is the property of some unstable chemical elements to emit very energetic radiations able to ionize and excite the matter. The emission of ionizing radiations is a characteristic of many atoms, called radioactive, in which the number of neutrons is insufficient or excessive, which it makes them unstable.

Radiation emission can be four different types: alpha (α), with limited capacity to penetrate matter but with very energetic intensity; beta (β), a little more penetrating but less intense; gamma (γ), very penetrating; and Neutronic, also very penetrating.

The intensity with which a radioactive substance is disintegrated is called Activity and expresses the number of atoms that are disintegrated per time unit, having as derived unit of the international system the Becquerel (Bq). It is also possible to use another activity unit, the Curie (Ci), named that way as a tribute to the physicists and chemists Pierre and Marie Curie. 1 Curie is equivalent to $3.7 \times 10^{10}$ Becquerels.
The equivalent dose is a physical magnitude that describes the relative effect of different types of ionizing radiation on living tissues. Its unit of measurement of the international system is the Sievert (Sv). Formerly it was used as a unit of equivalent dose, the Rem (Roentgen Equivalent Man) and the conversion between each unit is 1 Sievert equals 100 rem.

Exposure to radiation should be kept to a minimum because their effects are cumulative. For this reason, annual dose limits for the public and for workers who may be occupationally exposed are established. In Spain, the effective dose limit for members of the public will be 1 millisievert (mSv) per official year and the effective dose limit for skilled workers is 100 mSv throughout the consecutive period of five official years, subject to a maximum effective dose of 50 mSv in any official year.

Nuclear reactions are classified as nuclear fission reactions and nuclear fusion reactions. The most researched and developed process for massively obtaining useful energy from nuclear power is nuclear fission.

**Nuclear fission** is a reaction in which the neutrons are impinged on a heavy nucleus (usually uranium and plutonium), that is divided into two nuclei, generally radioactive and with random mass, emitting two or three neutrons and releasing a lot of energy. This energy is emitted in form of gamma radiation and kinetic energy of the fission fragments which heat the matter that is around the space where fission occurs.

Furthermore, emitted neutrons may cause new fissions in interacting with new fissionable nuclei that will emit new neutrons and so on. This multiplier effect is known as chain reaction.
The chain reaction is represented by the **multiplication factor** \((K_{\text{eff}})\), defined as the ratio between the number of fissions (or neutrons) of a given generation and the number of fissions (or neutrons) of the previous generation.

By **reactivity** \((\rho)\) is known the capacity of the reactor to multiply the neutron population, and is numerically defined as:

\[
\rho = \frac{K_{\text{eff}} - 1}{K_{\text{eff}}} \quad (F.1)
\]

Therefore, the chain reaction ceases if the multiplication factor \((K_{\text{eff}})\) is less than one and the reactivity \((\rho)\), less than zero. This means that the neutron population decreases and the reactor is **subcritical**. However, if \(K_{\text{eff}}\) is equal to unity and \(\rho\), equal to zero, the chain reaction is maintained. Consequently, the reactor is in **critical** condition and the neutron population remains constant. The critical term refers to a state of dynamic equilibrium in the fission chain reaction; in it there is no increase in power, temperature and neutron density in time. Finally, if \(K_{\text{eff}}\) exceeds unity and \(\rho\), greater than zero, the chain reaction spreads, being the reactor in **supercritical** state. In this situation, the neutron population increases.

To be able to control the reactivity in the water pressure reactors, **boron** is diluted in the water (as other forms to control reactivity) since it has a high ability to absorb neutrons, thereby decreasing the neutron population and extinguishing chain reaction.

However, once the reactor has stopped, large amount of heat continues to be released in the fuel due to the presence of decayed fission products (decay or disintegration heat).

This heat is called **residual heat or power** and it is necessary to evacuate it cooling it continuously to avoid damage to the fuel elements.
F.2 DESCRIPTION OF A PWR NUCLEAR PLANT

The main application of nuclear energy is the production of electricity in nuclear power plants. Therefore, in this section the operation of a pressurized water nuclear power plant or Pressurized Water Reactor (PWR) is briefly described because it is the type of plant on which this study is based. It is intended that with the following description the non-specialist in nuclear energy reader become familiar with the basics of the plant operation.

F.2.1 NUCLEAR PLANT DESCRIPTION

A nuclear power plant is an industrial facility used for the generation of electricity from nuclear energy, which is characterized by the use of fissile material that through nuclear reactions provide heat. This heat is used by a conventional thermodynamic steam cycle or Rankine to move a generator and produce electricity.

The type of reactor studied uses uranium dioxide (UO2) enriched as fuel and is cooled and moderated by light water at high pressure (hence the name Pressurized Water Reactor) and in liquid phase. It consists of three separate hydraulic circuits: primary, secondary and tertiary.

![Diagram of a pressurized water nuclear plant (PWR).](image)

The primary one consists of between 1 and 4 loops in parallel containing the water flowing through the core.
A pressurizer keeps the water at high pressure to reach high temperatures without boiling.

The primary circuit water, propelled by centrifugal pumps, goes through the core and removes the heat generated, transferring it after one or more exchangers or steam generators water at lower pressure in the secondary circuit. This lower pressure allows the water of the secondary circuit reaches saturation and boils in the side of the steam generator shell, producing steam that is conducted through a line to a turbine where it expands making them rotate at high speed.

The turbines spin an electric generator. Upon leaving the turbine, the steam condenses in the condenser and is pumped back to the steam generator.

The condenser exchanges heat with the tertiary circuit which evacuates heat from the plant to the last sink, usually the sea or river. In the case where the flow rate of tertiary circuit is insufficient, it is possible to install a cooling tower to lower the temperature of the cooling water used, avoiding an excessive heating of the water.

### F.2.2 OPERATION MODES

Plant operating states or modes of operation are different ways of operation of the plant under a combination of core reactivity, rated thermal input level and average reactor coolant temperature. In the Nuclear Plant studied, 6 modes are defined:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
<th>Keff</th>
<th>% Nominal Thermal Power*</th>
<th>Avgre temp of coolant</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OPERATION</td>
<td>≥ 0.99</td>
<td>&gt; 5%</td>
<td>≥ 175°C</td>
</tr>
<tr>
<td>2</td>
<td>START UP</td>
<td>≥ 0.99</td>
<td>≤ 5%</td>
<td>≥ 175°C</td>
</tr>
<tr>
<td>3</td>
<td>SHUTDOWN</td>
<td>&lt; 0.99</td>
<td>0</td>
<td>≥ 175°C</td>
</tr>
<tr>
<td>4</td>
<td>HOT SHUTDOWN</td>
<td>&lt; 0.99</td>
<td>0</td>
<td>175°C &gt; T &gt; 93°C</td>
</tr>
<tr>
<td>5</td>
<td>COLD SHUTDOWN</td>
<td>&lt; 0.99</td>
<td>0</td>
<td>≤ 93°C</td>
</tr>
<tr>
<td>6</td>
<td>RELOAD**</td>
<td>≤ 0.95</td>
<td>0</td>
<td>≤ 60°C</td>
</tr>
</tbody>
</table>

For each mode of operation, the plant may have different features or specific configurations to subdivide that mode in various operational states of Plant (EOPs).
ENVIRONMENTAL IMPACT
ENVIROMENTAL IMPACT

The actual study develops a safety assessment of a nuclear power plant by performing computer simulations. Hence, the offered service consist of engineering safety assessment to the plant.

No physical modification of any system, component or material of the plant is required as a consequence of this study, which as a result provides the plant with a better understanding of general risks about external events being analysed.

For this reason, the environmental impact evaluation in this project focuses on qualitative balance between consumed energy by the computers used to perform the simulations, and the environmental impact that a rare event leading to an accident on the plant would cause.

On the one hand, the results of the actual study allow the improvement of the safety management of generic nuclear power plants. This is done by assessing the control room habitability in a case of external toxic gas dispersion coming from a train accident, by means of hazardous materials train shipping.

On the other hand, the fact that a nuclear power plant may improve the understanding of external risks, has more weight on positive environmental impact than that of consumed electricity by the computers used for this purpose negatively have.

Therefore, it is safe to say that this study has positive impact on the environment.
COST ANALYSIS
COST ANALYSIS

1. RESOURCES DISTRIBUTION

The required budget for the development of this project consist of two kind of costs. Firstly the labour costs as engineering hours, and secondly material resources costs.

Engineering hours are classified in two types:

- Senior engineer. The project manager, as senior engineer, takes the supervision and the direction of the study, as well as the coordination of the meetings with the staff of the plant.
- Junior engineer, which carries with the main work of the project, under the manager supervision.

The material resources are divided in the following categories:

- Facilities. This project has been developed in the UPC facilities. No costs are considered by this category.
- Hardware. The equipment used for this study consist of one personal computer which belongs to Polytechnic University of Catalonia, and one laptop that belongs to private user. The equipment depreciation is considered in the costs analysis.
- Software. The used programming languages are open, as well as the NEA codes, and some other non-free applications have been used as student version. Hence, no costs are assigned to software.
- Traveling costs. The agreement with the plant has been accomplished by one meeting in the site. So transportation costs are to be considered.
- Printing and binding.
2. LABOUR COSTS QUANTIFICATION

The national wages regulations establish the quantities to assign depending on educational level. In this case, the Master’s degree is classified as follows:

<table>
<thead>
<tr>
<th>Level 1. Master’s Degree</th>
<th>Month $\times 14$</th>
<th>Annual €</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1687.02</td>
<td>23618.28</td>
</tr>
</tbody>
</table>

The basis for public assurance contribution (“seguirad social”) are defined as:

<table>
<thead>
<tr>
<th>Grupo de Cotización</th>
<th>Categorías Profesionales</th>
<th>Bases mínimas €/mes</th>
<th>Bases máximas €/mes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ingenieros y Licenciados. Personal de alta dirección no incluido en el artículo 1.3.c) del Estatuto de los Trabajadores</td>
<td>1056.9</td>
<td>3606</td>
</tr>
</tbody>
</table>

On the account that the Senior engineer has an increase of 40% in salary, the labour costs are calculated. The result is presented in (Eur/hour)

<table>
<thead>
<tr>
<th></th>
<th>€/h</th>
</tr>
</thead>
<tbody>
<tr>
<td>Senior engineer</td>
<td>25.94</td>
</tr>
<tr>
<td>Junior engineer</td>
<td>18.54</td>
</tr>
</tbody>
</table>

The spent time for developing this project is described in the next table

<table>
<thead>
<tr>
<th></th>
<th>months</th>
<th>h/month</th>
<th>Meeting h/t.</th>
<th>Number.</th>
<th>total hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Junior engineer</td>
<td>5</td>
<td>148</td>
<td>6</td>
<td>1</td>
<td>746</td>
</tr>
<tr>
<td>Project manager (senior)</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td>36</td>
</tr>
</tbody>
</table>

Finally, the labour cost before taxes is 14.764,68 €
3. COST OF MATERIAL RESOURCES

The maximum depreciation for computer equipment is 25% a year, and it is assigned a maximum period of 8 years.

The NERG(UPC) equipment was purchased in 2013 for a 1200 € price. A depreciation of 25% is applied finishing the total period. Thus, the assigned cost is 300 €.

The private equipment was bought in 2014 for a 700 € price. A depreciation of 25% is applied returning a cost of 175 €.

Other costs are described next:

<table>
<thead>
<tr>
<th>Table 1.5. Material resources costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Printing</td>
</tr>
<tr>
<td>Binding</td>
</tr>
<tr>
<td>CDs</td>
</tr>
<tr>
<td>Transport</td>
</tr>
</tbody>
</table>

4. TOTAL COSTS TABLE

<table>
<thead>
<tr>
<th>Table 1.6. Total costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Material costs €</td>
</tr>
<tr>
<td>Transport €</td>
</tr>
<tr>
<td>Labour costs €</td>
</tr>
<tr>
<td>Total</td>
</tr>
<tr>
<td>Total + IVA (21%)</td>
</tr>
</tbody>
</table>
### PROJECT GANTT DIAGRAM

<table>
<thead>
<tr>
<th>ID</th>
<th>Task Description</th>
<th>Start Date</th>
<th>End Date</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Problem Analysis and Research</td>
<td>15/12/2015</td>
<td>21/01/2016</td>
<td>28d</td>
</tr>
<tr>
<td>2</td>
<td>First meeting with licensee</td>
<td>27/01/2016</td>
<td>27/01/2016</td>
<td>0d</td>
</tr>
<tr>
<td>3</td>
<td>Dispersion codes familiarization</td>
<td>28/01/2016</td>
<td>04/02/2016</td>
<td>6d</td>
</tr>
<tr>
<td>4</td>
<td>Alternative dispersion models research</td>
<td>28/01/2016</td>
<td>16/02/2016</td>
<td>14d</td>
</tr>
<tr>
<td>5</td>
<td>Meteorology familiarization</td>
<td>04/02/2016</td>
<td>07/07/2016</td>
<td>14d</td>
</tr>
<tr>
<td>6</td>
<td>Programming tools research</td>
<td>23/02/2016</td>
<td>18/03/2016</td>
<td>19d</td>
</tr>
<tr>
<td>7</td>
<td>Model developing</td>
<td>21/03/2016</td>
<td>27/04/2016</td>
<td>28d</td>
</tr>
<tr>
<td>8</td>
<td>Second meeting (telematic)</td>
<td>11/04/2016</td>
<td>11/04/2016</td>
<td>0d</td>
</tr>
<tr>
<td>9</td>
<td>Documents writing</td>
<td>11/04/2016</td>
<td>30/05/2016</td>
<td>36d</td>
</tr>
<tr>
<td>10</td>
<td>Delivering</td>
<td>07/06/2016</td>
<td>07/06/2016</td>
<td>0d</td>
</tr>
</tbody>
</table>

---

**Research period**

**Solution Development**